
# *AN ANALYSIS OF BOGON IP TRAFFIC IN THE SURFNET6 NETWORK*

SURFnet6 Research on Networks

SARA Network Research Group <nrg@sara.nl>

March 2007

SARA High-Performance Networking, Amsterdam.
Address:
Kruislaan 415
1098 SJ Amsterdam
Tel.: +31 20  592 3000
Fax: +31 20  668 3167

Andree Toonk
Andree@sara.nl
Tel: +31 20 5928000

## ACKNOWLEDGEMENT

The author would like to thank Ronald van der Pol and Marco Davids for their excellent suggestions. The SARA Network Research Group for the discussions on this topic. Hans Trompert of SURFnet for his support while building the netflow setup. Bas Toonk for his help with writing the Spamassassin plugin.

# INDEX

# INTRODUCTION

Bogons are IP prefixes that should never appear in the Internet routing table, and obviously should not appear in any packets in the network. During this project we have searched for bogon IP addresses in the SURFnet6 IP network.

It is common practice to apply outbound filters on a BGP peering with external BGP peers, this is done to prevent the local AS from leaking prefixes other then the prefixes which are assigned to the local AS. The same filtering technique can also be applied for inbound filtering. This is done to protect against learning prefixes which are not assigned to the origin AS advertising the prefixes. Building these inbound filters is time consuming and these filters will also change daily which makes it even more time consuming. As a result a lot of Internet providers do not have strict filters. This gives people the opportunity to advertise any bogon prefix they want.
The definition of a bogon IP address is an IP address or a range addresses which is not yet assigned to a RIR, these ranges are still in the IANA reserved range pool. Most of them are for future use or have a special purpose, such as RFC1918 space or link local addresses. During this project the netflow data of the SURFnet6 IP network has been analyzed to see if and what kind of bogon IP traffic is routed over the network.

This document describes the results of the analyses of the netflow data captured during this project. In addition to the netflow analyses an analysis of bogon IP addresses in email headers and the global routing table was done. This project was performed as part of the Research on Networks GigaPort project.

# ABOUT THE SURFNET6 NETWORK

SURFnet is the Dutch national research and education network and provides its users with highly innovative Internet services. SURFnet6, the sixth generation of the SURFnet network, is a hybrid network offering both IP and optical services. The SURFnet6 network was taken into operation at the beginning of 2006.

The IP core is built with highly scalable Avici routers. The SURFnet6 IP network offers both IPv4 and IPV6 unicast and multicast Internet connectivity. The IP network is built on top of the optical network layer.
This optical infrastructure consists of Nortel equipment and 6000km fiber on which SURFnet has the long-term rights of use. In addition to IP connectivity SURFnet6 also offers lightpath services. Lightpaths are point to point connections and are characterized by a very high capacity and a deterministic behavior.

SURFnet6 was developed in the GigaPort project, which runs from 2004 until 2008.

## WHAT IS A BOGON

A good description of a bogon is the definition below which was originally mentioned in *Hunting the Bogon*[1]:

***The use of an address or, more generally a route object, that is not authorized by the entity to which the address, or resource, was originally assigned to.***

According to the definition above these are a few examples of bogons:

- The resource has never been allocated (still in the IANA[2]/RIR pool)
- The resource has been hijacked
- Incorrect use of special/reserved addresses (RFC 1918 / RFC 3330 )

A resource can be:

- an IP address or an IP range
- a AS number

It is important to have a good understanding of the bogon definition used in this project. IP ranges which are still in the pool of the RIR's or LIR's as well as AS numbers are not taken into account during this project.
During this project IPv4 prefixes are considered as a bogon when they are reserved prefixes or prefixes which are still in the IANA pool and not yet allocated to a RIR.. The IPv4 bogon list provided by team Cymru is used to filter out the IPv4 bogons. This list can be found on: http://www.cymru.com/Documents/bogon-bn-nonagg.txt.
For the IPv6 bogons the bogons definition is stricter. The recommendations on http://www.space.net/~gert/RIPE/ipv6-filters.html were used. This list defines which prefixes are assigned to the RIRs. Every other prefix we consider as a bogon.

[1] Hunting the Bogon; Geoff Huston, The ISP Column, May 2004

[2] http://www.iana.com/

## NETFLOW DATA COLLECTION

The Avici core routers export sampled (1/100) netflow (version 9) data to the data collectors of SURFnet. For this project the netflow streams were replicated to our local (SARA) analyzing system. This was done over a secure connection using Zebedee[3].
Flowd[4] is used as a netflow server, this server will receive the netflow data and stores it on disk. The netflow data is multiple gigabytes each day. The majority of this data is not interesting for this project because this is non-bogon data. Only the netflow data which contains a bogon source or destination IPv4 or IPv6 address is saved on disk.
The filtering of the relevant data is done at the SARA netflow collector. Two filter lists are used, one for IPv4 and one for IPv6 bogons. The IPv4 filter list is created using the bogon list on: http://www.cymru.com/Documents/bogon-bn-nonagg.txt. The IPv6 filter is made using the recommendations found on: http://www.space.net/~gert/RIPE/ipv6-filters.html
The complete filters can be found in Appendix I.

Multiple datasets are used during this project, each dataset contains one week of bogon netflow data. Each dataset is inserted in a MySQL databases so it can be queried easily when analyzing the data. See figure 1 for the complete setup

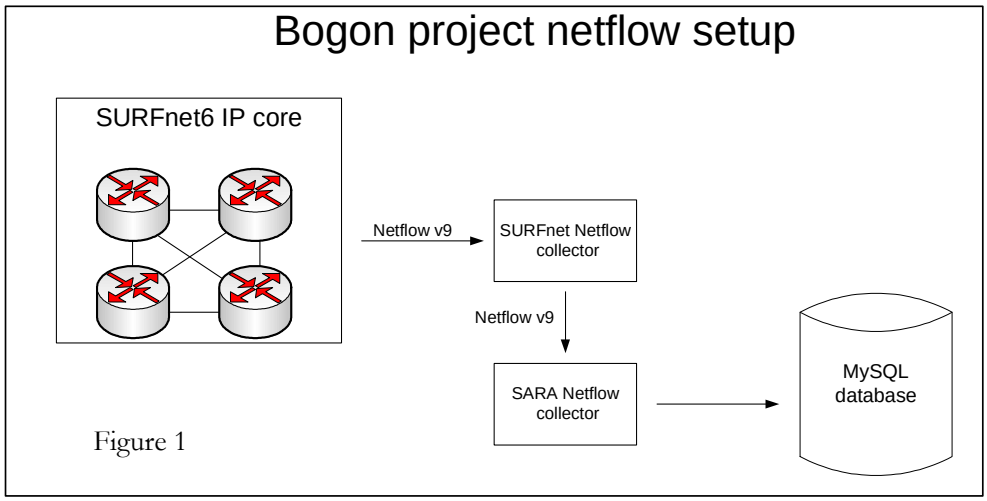



Figure 1

The total amount of bogon flows analyzed during this project is 86.271.118 flows. Only 2906 of these bogon flows are IPv6 flows.

[3] http://www.winton.org.uk/zebedee/

[4] http://www.mindrot.org/projects/flowd/

# IPV4 ANALYSIS

In this chapter different top 10 lists are listed, these top 10 lists are generated by querying the database. Those top 10 lists will then be further analyzed.

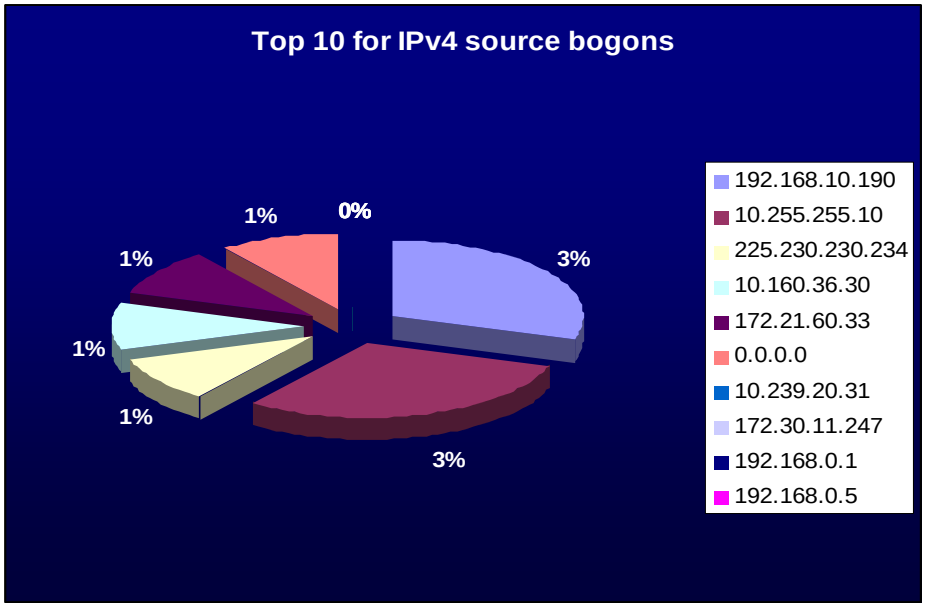
First we will take a look at the top 10 of different bogon addresses. Four types of bogons are defined:

1. Bogon is an IPv6 source address
2. Bogon is an IPv6 destination address
3. Bogon is an IPv4 source address
4. Bogon is an IPv4 destination address

Below the top 10 of the IPv4 groups of bogons are shown.

## IPV4 SOURCE BOGONS:

| **total inet4 src bogons** | 879346 | |
|---|---|---|
| 192.168.10.190 | 28496 | 3% |
| 10.255.255.10 | 24742 | 3% |
| 225.230.230.234 | 11283 | 1% |
| 10.160.36.30 | 9303 | 1% |
| 172.21.60.33 | 8935 | 1% |
| 0.0.0.0 | 5629 | 1% |
| 10.239.20.31 | 3453 | 0% |
| 172.30.11.247 | 3436 | 0% |
| 192.168.0.1 | 3436 | 0% |
| 192.168.0.5 | 3189 | 0% |



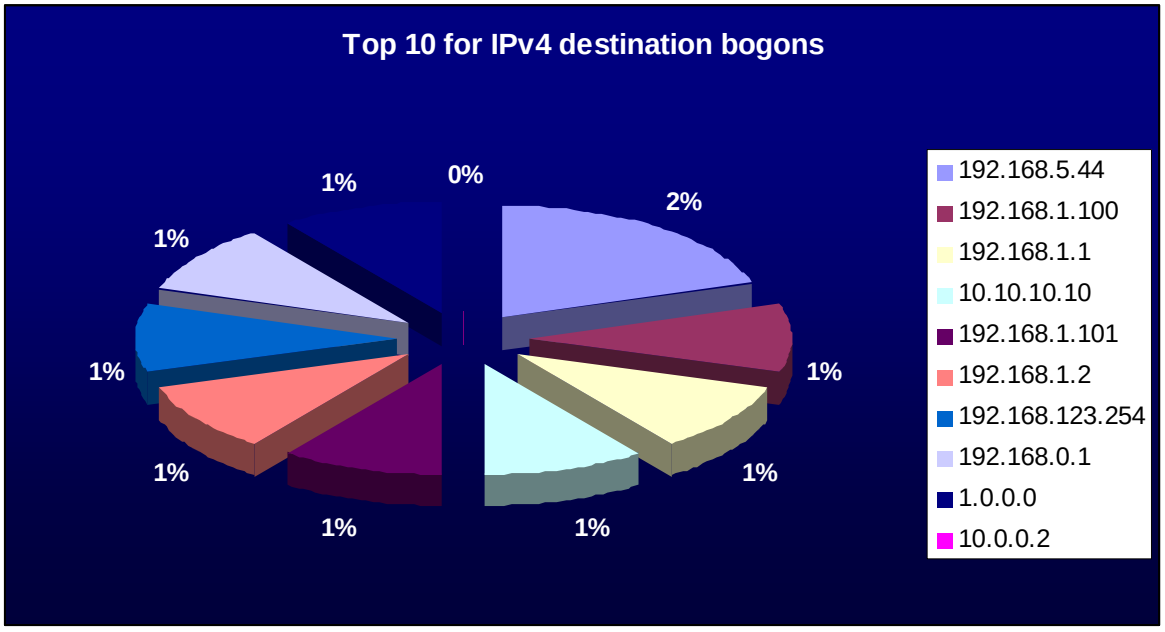
The table below represents the top source and destination ports in the flows where the IPv4 source address is a bogon.

| proto | srcport |
|---|---|
| TCP | 3072 |
| TCP | 1024 |
| ICMP | 0 |
| UDP | 1026 |
| UDP | 1027 |
| TCP | 3002 |
| TCP | 11083 |
| UDP | 137 |
| UDP | 45604 |
| TCP | 25 |

| Proto | dstport |
|---|---|
| UDP | 1026 |
| TCP | 80 |
| TCP | 22 |
| ICMP | 0 |
| UDP | 137 |
| UDP | 53 |
| TCP | 445 |
| TCP | 337 |
| UDP | 1434 |
| TCP | 135 |

## IPV4 DESTINATION BOGONS

| total inet4 dst bogons | 23262362 | |
|---|---|---|
| 192.168.5.44 | 449408 | 2% |
| 192.168.1.100 | 287108 | 1% |
| 192.168.1.1 | 203720 | 1% |
| 10.10.10.10 | 201524 | 1% |
| 192.168.1.101 | 190638 | 1% |
| 192.168.1.2 | 156340 | 1% |
| 192.168.123.254 | 153707 | 1% |
| 192.168.0.1 | 151544 | 1% |
| 1.0.0.0 | 117088 | 1% |
| 10.0.0.2 | 115684 | 0% |

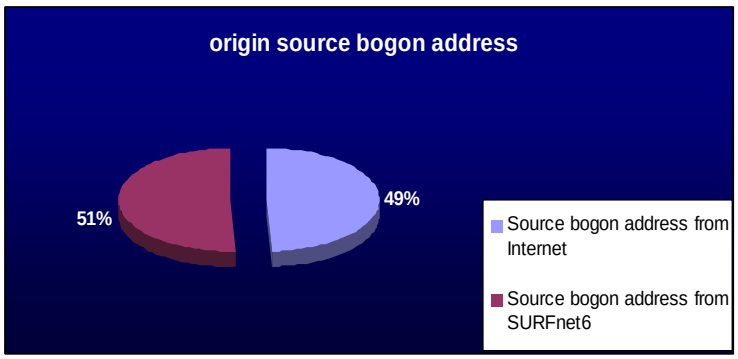


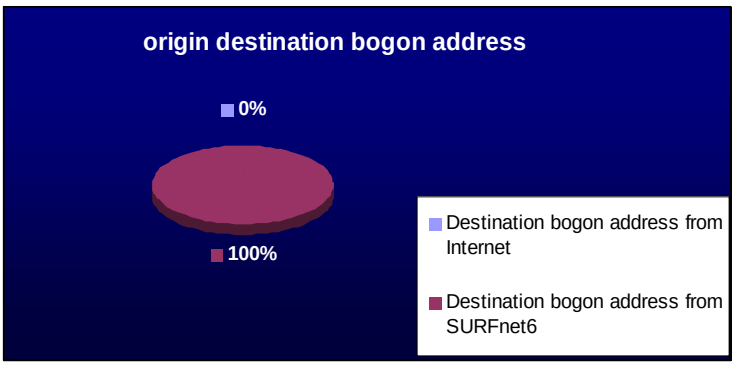
The table below represents the top destination and source ports in the flows where the IPv4 destination address is a bogon.

| proto | srcport |
|---|---|
| ICMP | 0 |
| UDP | 123 |
| TCP | 80 |
| UDP | 1230 |
| UDP | 3098 |
| UDP | 10879 |
| UDP | 8841 |
| UDP | 137 |
| UDP | 3853 |
| UDP | 5284 |

| proto | dstport |
|---|---|
| TCP | 5900 |
| ICMP | 0 |
| TCP | 80 |
| UDP | 38293 |
| UDP | 1230 |
| UDP | 161 |
| TCP | 4662 |
| UDP | 123 |
| UDP | 53 |
| TCP | 82 |

There are significantly more IPv4 destination bogons then IPv4 source bogons. The graphs below show that the majority of the IPv4 bogon flows with a destination bogon address are originated from SURFnet6 costumers. There is almost no bogon traffic coming in from outside SURFnet6 to the SURFnet6 network with a destination bogon address.
For IPv4 bogon source addresses the origin proportion is almost equal. 51% of the flows with a bogon source address is originated from SURFnet costumers, 49% comes from outside the SURFnet6 network.



## EXCLUDE RFC1918 SPACE

It is clearly visible in the top 10 list of the IPv4 destination bogons, that the majority is RFC 1918 space. In fact 80% of the all the flows with a destination bogon address is an address from the RFC1918 address space.
When analyzing the flows with an IPv4 source bogons address, we see that 31% of the total amount these flows is RFC 1918 space. According to the best current practices these IP addresses should not be seen in the core of SURFnet6. It is however well known these addresses are visible when not applying filters. Therefore it is interesting to see which bogons, besides the RFC 1918 addresses are visible. Below an overview of the top 10 source and destination bogon addresses excluding RFC1918 addresses.

| **total inet4 src bogons (non RFC1918)** | 609983 | |
|---|---|---|
| 225.230.230.234 | 11283 | 1,8497% |
| 0.0.0.0 | 5629 | 0.9221% |
| 102.139.220.86 | 2532 | 0.4150% |
| 103.98.49.43 | 2082 | 0.3413% |
| 169.254.25.129 | 724 | 0.1186% |
| 95.15.183.16 | 512 | 0.0839% |
| 113.243.164.146 | 480 | 0.0786% |
| 187.158.237.73 | 479 | 0.07852% |
| 95.228.151.137 | 468 | 0.07672% |
| 169.254.27.20 | 370 | 0.06065% |

| **total inet4 dst bogons (non RFC1918)** | 4700343 | |
|---|---|---|
| 1.0.0.0 | 117088 | 2,49105224874015% |
| 1.1.1.2 | 66024 | 1,40466344690164% |
| 112.5.145.124 | 63385 | 1,34851860811009% |
| 1.1.1.1 | 59417 | 1,26409923701313% |
| 169.254.2.2 | 36999 | 0,787155320367045% |
| 234.5.6.7 | 33180 | 0,705905930694845% |
| 112.5.144.124 | 18576 | 0,395205200982141% |
| 39.123.0.0 | 17837 | 0,379482944117057% |
| 169.254.254.229 | 16962 | 0,360867281387763% |
| 169.254.255.255 | 14366 | 0,305637269450336% |

When ignoring the RFC1918 address space, there are still 609.983 entries which have an IPv4 bogon source address and 4.700.343 (four and a half million) an IPv4 bogon destination address.

Let's take a deeper look into a few of them.

### *225.230.230.234*

According to RFC 3171 (IANA Guidelines for IPv4 Multicast Address Assignments) 225.0.0.0 - 231.255.255.255 is a reserved range and therefore should never be visible. Besides that, multicast addresses should never be visible as source addresses. Further analyses show that all the flows have the same traffic properties. It is all TCP traffic originating from port 13364 to different destination addresses with the same destination ports: 139, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034 and 1035.

### *0.0.0.0*

0.0.0.0 is a special kind of address; normally it can be seen in a local network when a host is in the process of acquiring an IPv4 address using DHCP or BOOTP.
Analyzing all the bogon data which has 0.0.0.0 as source address show that 99% of this traffic is from one SURFnet6 customer's firewall cluster, with the following properties: TCP: 0.0.0.0: 8116 <> 145.145.xx.xx port8116. This is probably because of a configuration error at the customer side.

Next we see ICMP traffic from 0.0.0.0 to different IPv4 unicast addresses. It is not clear what kind of traffic that is. On one the core routers we also see some DHCP traffic, however these are only 9 flows in one week. Nevertheless these flows should not be visible because of an anti-spoofing filter.

### *102.139.220.86 AND 103.98.49.43*

According to IANA the addresses 102.139.220.86 and 103.98.49.43 are reserved and should therefore not be visible.

```
NetRange:   100.0.0.0 - 120.255.255.255
CIDR:       100.0.0.0/6, 104.0.0.0/5, 112.0.0.0/5, 120.0.0.0/8
NetName:    RESERVED-8
NetHandle:  NET-100-0-0-0-1
Parent:
NetType:    IANA Reserved
```

Currently both of these addresses are not reachable and are not present in the routing table (anymore). When looking for this traffic in the database, it shows that all the traffic is UDP and has a range of destination addresses within one university. The source port and the range of destination ports are always the same. See the table below for the numbers.

| proto | srcip | srcport | dstport | count |
|---|---|---|---|---|
| 17 | 102.139.220.86 | 45604 | 1027 | 524 |
| 17 | 102.139.220.86 | 45604 | 1025 | 516 |
| 17 | 102.139.220.86 | 45604 | 1026 | 513 |
| 17 | 102.139.220.86 | 45604 | 1028 | 499 |
| 17 | 102.139.220.86 | 45604 | 1029 | 480 |
| 17 | 103.98.49.43 | 45604 | 1027 | 449 |
| 17 | 103.98.49.43 | 45604 | 1025 | 418 |
| 17 | 103.98.49.43 | 45604 | 1029 | 416 |
| 17 | 103.98.49.43 | 45604 | 1026 | 403 |
| 17 | 103.98.49.43 | 45604 | 1028 | 396 |

This seems to be a scan to hosts in that specific university network to some well known Windows UDP ports. These specific ports are often used by viruses and worms.

### *169.254.25.129 AND 169.254.27.20*

These addresses are from the IP range 169.254.0.0/16 - This is the "link local" block. It is allocated for communication between hosts on a single link. Hosts obtain these addresses by auto-configuration, such as when a DHCP server may not be found, this is used by Windows hosts and Microsoft calls this Automatic Private IP Addressing (APIPA).

| proto | srcip | srcport | dstip | dstport | count |
|---|---|---|---|---|---|
| 17 | 169.254.25.129 | 123 | NTP server of University X | 123 | 715 |
| 6 | 169.254.25.129 | 19634 | University X | 51463 | 3 |
| 17 | 169.254.25.129 | 137 | 192.168.1.19 | 137 | 2 |
| 6 | 169.254.25.129 | 1401 | 192.168.1.19 | 135 | 2 |
| 6 | 169.254.25.129 | 1459 | 192.168.1.19 | 135 | 1 |
| 6 | 169.254.25.129 | 5407 | 192.168.1.19 | 135 | 1 |

Most of these flows go the NTP server of a university in The Netherlands. Those might be hosts which did not get an address using DHCP and try to synchronize their clock with a NTP server.

### *113.243.164.146 AND 187.158.237.73 AND 95.228.151.137 AND 95.15.183.16*

| proto | srcip | srcport | dstport | count |
|---|---|---|---|---|
| 17 | 187.158.237.73 | 45604 | 1029 | 111 |
| 17 | 113.243.164.146 | 45604 | 1025 | 110 |
| 17 | 95.15.183.16 | 45604 | 1025 | 110 |
| 17 | 95.228.151.137 | 45604 | 1028 | 109 |
| 17 | 95.15.183.16 | 45604 | 1029 | 108 |
| 17 | 95.15.183.16 | 45604 | 1027 | 103 |
| 17 | 113.243.164.146 | 45604 | 1026 | 101 |
| 17 | 187.158.237.73 | 45604 | 1025 | 99 |
| 17 | 95.228.151.137 | 45604 | 1025 | 98 |
| 17 | 95.15.183.16 | 45604 | 1026 | 98 |
| 17 | 187.158.237.73 | 45604 | 1027 | 96 |
| 17 | 113.243.164.146 | 45604 | 1028 | 96 |
| 17 | 187.158.237.73 | 45604 | 1026 | 94 |
| 17 | 95.15.183.16 | 45604 | 1028 | 93 |
| 17 | 95.228.151.137 | 45604 | 1029 | 88 |
| 17 | 95.228.151.137 | 45604 | 1026 | 88 |
| 17 | 113.243.164.146 | 45604 | 1029 | 87 |
| 17 | 113.243.164.146 | 45604 | 1027 | 86 |
| 17 | 95.228.151.137 | 45604 | 1027 | 85 |
| 17 | 187.158.237.73 | 45604 | 1028 | 79 |

The destination IP addresses are all in the same /15 of one of the SURFnet customers. These flows look like scans to well known Microsoft Windows services. The Source address 95.15.183.16 is currently not visible in the SURFnet6 routing tables. The scans are all to one customer range, most of these hosts reply with a ICMP message.

### *1.0.0.0*

The destination IP address 1.0.0.0 is the number 1 in the top 10 of IPv4 destination bogons. Lets take a closer look in what kind of traffic this is. Below is a table of used destination ports.

| proto | dstip | dstport | count |
|---|---|---|---|
| 17 | 1.0.0.0 | 0 | 102386 |
| 17 | 1.0.0.0 | 832 | 3092 |
| 17 | 1.0.0.0 | 2052 | 1312 |
| 6 | 1.0.0.0 | 80 | 322 |
| 17 | 1.0.0.0 | 16900 | 253 |
| 17 | 1.0.0.0 | 51459 | 210 |
| 17 | 1.0.0.0 | 51203 | 146 |
| 17 | 1.0.0.0 | 21763 | 109 |
| 17 | 1.0.0.0 | 20739 | 106 |
| 17 | 1.0.0.0 | 21251 | 86 |

The first entry of this table is remarkable. Not only is the destination IP address bogon, but so is the destination port (zero). The next step is to take a closer look into this traffic. In the table below the source IP address is also taken into account, this is to make sure that the high number is not caused by one host. The IP addresses have been rewritten to xxxx, every xxxx is a unique IP address. This top 20 of the unique combination proto, source IP, destination IP and destination port, show that there are different hosts which all sent the same kind of traffic to 1.0.0.0 port 0. There is one host in this top 20 which sends a lot flows to destination port 832, this port is assigned to netconfsoaphttp service.

| proto | srcip | dstip | dstport | count |
|---|---|---|---|---|
| 17 | xxxx | 1.0.0.0 | 0 | 27443 |
| 17 | xxxx | 1.0.0.0 | 0 | 18344 |
| 17 | xxxx | 1.0.0.0 | 0 | 6499 |
| 17 | xxxx | 1.0.0.0 | 0 | 5800 |
| 17 | xxxx | 1.0.0.0 | 0 | 5066 |
| 17 | xxxx | 1.0.0.0 | 0 | 4181 |
| 17 | xxxx | 1.0.0.0 | 0 | 3374 |
| 17 | xxxx | 1.0.0.0 | 832 | 3092 |
| 17 | xxxx | 1.0.0.0 | 0 | 3084 |
| 17 | xxxx | 1.0.0.0 | 0 | 2434 |
| 17 | xxxx | 1.0.0.0 | 0 | 2025 |
| 17 | xxxx | 1.0.0.0 | 0 | 1846 |
| 17 | xxxx | 1.0.0.0 | 0 | 1716 |
| 17 | xxxx | 1.0.0.0 | 0 | 1416 |
| 17 | xxxx | 1.0.0.0 | 0 | 1238 |
| 17 | xxxx | 1.0.0.0 | 0 | 739 |
| 17 | xxxx | 1.0.0.0 | 0 | 677 |
| 17 | xxxx | 1.0.0.0 | 0 | 614 |
| 17 | xxxx | 1.0.0.0 | 0 | 613 |
| 17 | xxxx | 1.0.0.0 | 0 | 604 |

Port 0 is officially a reserved port in TCP/IP networking, meaning that it should not be used for any TCP or UDP network communications.

### *1.1.1.2*

| proto | dstip | dstport | count |
|---|---|---|---|
| 6 | 1.1.1.2 | 6001 | 58473 |
| 6 | 1.1.1.2 | 20201 | 6033 |
| 1 | 1.1.1.2 | 0 | 1195 |
| 17 | 1.1.1.2 | 11719 | 107 |
| 17 | 1.1.1.2 | 3963 | 48 |
| 17 | 1.1.1.2 | 3251 | 35 |
| 17 | 1.1.1.2 | 3108 | 23 |
| 17 | 1.1.1.2 | 4581 | 18 |
| 17 | 1.1.1.2 | 1644 | 17 |
| 17 | 1.1.1.2 | 44674 | 15 |

About 89% of the flows toward this bogon IP address come from 2 distinct hosts in one /23 subnet. These hosts try to connect to 1.1.1.2 port 6001 this is X11 (X Window System port).

### *112.5.145.124*

| proto | dstip | dstport | count |
|---|---|---|---|
| 17 | 112.5.145.124 | 65535 | 62555 |
| 17 | 112.5.145.124 | 2052 | 70 |
| 17 | 112.5.145.124 | 51459 | 21 |
| 17 | 112.5.145.124 | 16900 | 16 |
| 17 | 112.5.145.124 | 17923 | 15 |
| 17 | 112.5.145.124 | 20739 | 13 |
| 17 | 112.5.145.124 | 21251 | 13 |
| 17 | 112.5.145.124 | 261 | 12 |
| 17 | 112.5.145.124 | 21763 | 12 |
| 17 | 112.5.145.124 | 48387 | 11 |

Almost all of the bogon flows with destination IP address of 112.5.145.124 have udp port 65535 as the destination port. Port number 65535 is the highest port number possible.
If we look at the next table, it is clear that there is a range of IP address generating the flows towards 112.5.145.124 port 65535. These IP addresses are from different customers within the SURFnet6 network.

| proto | srcip | dstip | dstport | count |
|---|---|---|---|---|
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 21613 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 7420 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 2917 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 2667 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 1892 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 1768 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 1409 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 1382 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 1314 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 1012 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 959 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 888 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 873 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 857 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 820 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 735 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 732 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 718 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 678 |
| 17 | x.x.x.x | 112.5.145.124 | 65535 | 492 |

### *1.1.1.1*

The majority of the flows towards 1.1.1.1 have destination port UDP port 6346. This port is often used by filesharing programs such as Limewire.

| proto | dstip | dstport | count |
|---|---|---|---|
| 17 | 1.1.1.1 | 6346 | 40256 |
| 6 | 1.1.1.1 | 20201 | 6092 |
| 6 | 1.1.1.1 | 2000 | 2931 |
| 6 | 1.1.1.1 | 25 | 2663 |
| 17 | 1.1.1.1 | 53 | 2609 |
| 6 | 1.1.1.1 | 80 | 2039 |
| 6 | 1.1.1.1 | 6346 | 1219 |
| 1 | 1.1.1.1 | 0 | 1120 |
| 6 | 1.1.1.1 | 8088 | 124 |
| 17 | 1.1.1.1 | 161 | 88 |

The table below shows the top 20 of distinct source IP addresses which had flows to 1.1.1.1.

| proto | srcip | dstip | dstport | count |
|---|---|---|---|---|
| 6 | x.x.x.x | 1.1.1.1 | 20201 | 6092 |
| 6 | x.x.x.x | 1.1.1.1 | 2000 | 2931 |
| 17 | x.x.x.x | 1.1.1.1 | 53 | 2587 |
| 6 | x.x.x.x | 1.1.1.1 | 80 | 1768 |
| 6 | x.x.x.x | 1.1.1.1 | 25 | 666 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 632 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 628 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 528 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 501 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 483 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 447 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 411 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 397 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 396 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 366 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 353 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 345 |
| 17 | x.x.x.x | 1.1.1.1 | 6346 | 344 |
| 6 | x.x.x.x | 1.1.1.1 | 25 | 320 |
| 6 | x.x.x.x | 1.1.1.1 | 25 | 318 |

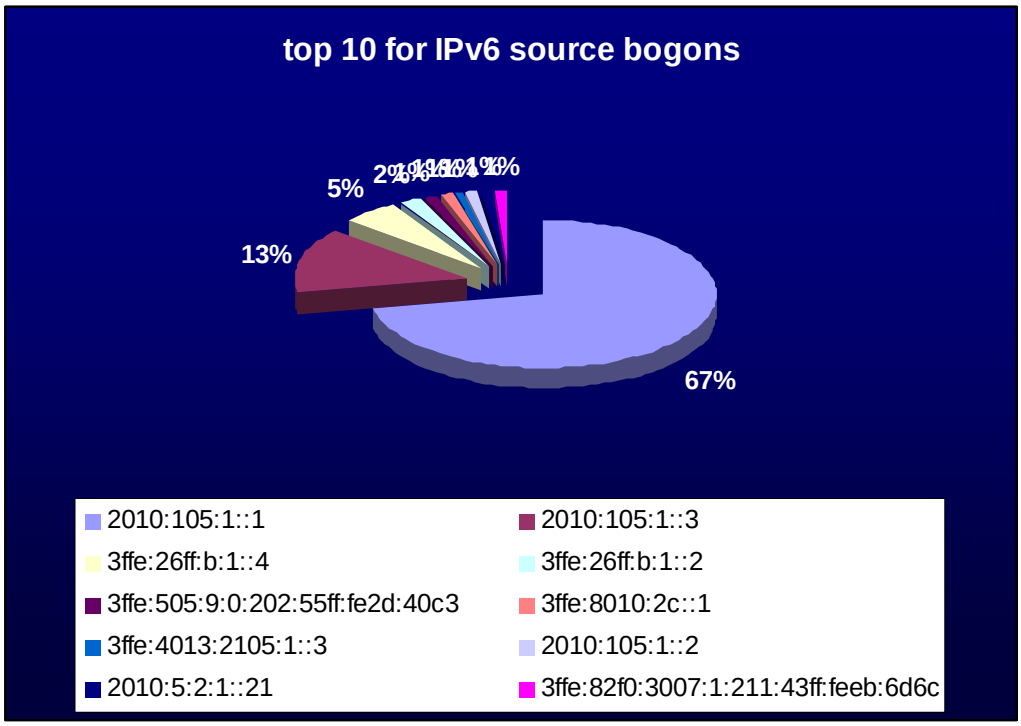
At the time these flows were analyzed they were not in the routing table; it could very well be that they were never there and that these flows are the cause of some kind of misconfiguration.

## IPV6 ANALYSES

In this chapter we will take a closer look at to the IPv6 bogons found during this project. Below are the top 10 IPv6 bogons.

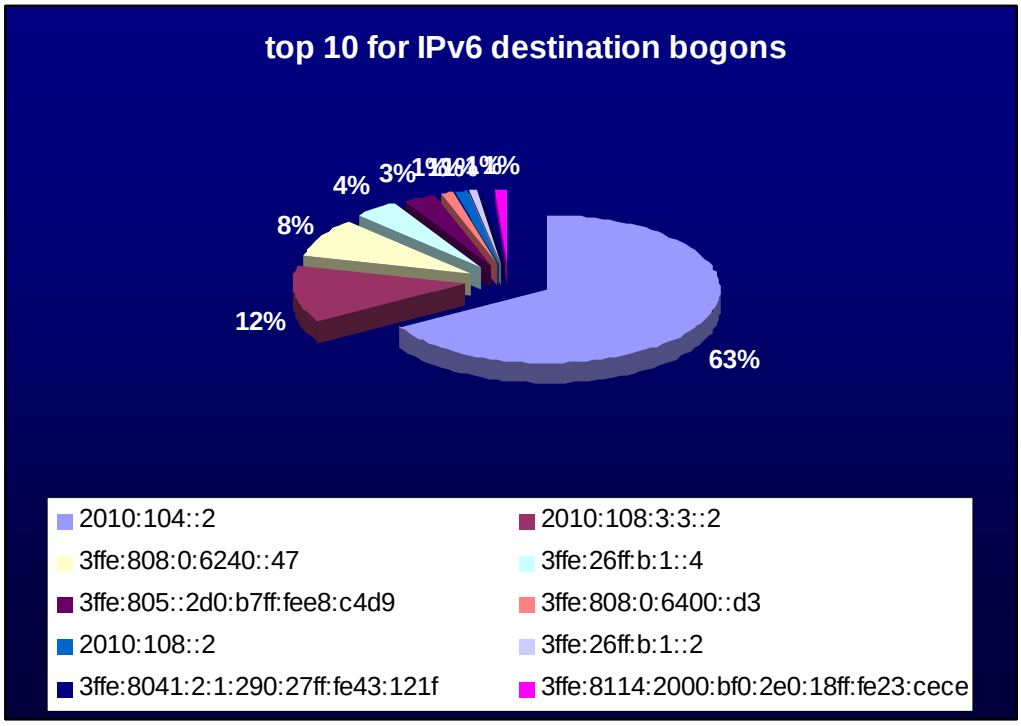
### IPV6 SOURCE BOGONS:

| **total inet6 src bogons** | **331** | |
|---|---|---|
| 2010:105:1::1 | 222 | 67% |
| 2010:105:1::3 | 42 | 13% |
| 3ffe:26ff:b:1::4 | 16 | 5% |
| 3ffe:26ff:b:1::2 | 8 | 2% |
| 3ffe:505:9:0:202:55ff:fe2d:40c3 | 4 | 1% |
| 3ffe:8010:2c::1 | 4 | 1% |
| 3ffe:4013:2105:1::3 | 4 | 1% |
| 2010:105:1::2 | 4 | 1% |
| 2010:5:2:1::21 | 3 | 1% |
| 3ffe:82f0:3007:1:211:43ff:feeb:6d6c | 3 | 1% |



## IPV6 DESTINATION BOGONS:

| total inet6 dst bogons | 354 | |
|---|---|---|
| 2010:104::2 | 222 | 63% |
| 2010:108:3:3::2 | 42 | 12% |
| 3ffe:808:0:6240::47 | 29 | 8% |
| 3ffe:26ff:b:1::4 | 15 | 4% |
| 3ffe:805::2d0:b7ff:fee8:c4d9 | 11 | 3% |
| 3ffe:808:0:6400::d3 | 4 | 1% |
| 2010:108::2 | 4 | 1% |
| 3ffe:26ff:b:1::2 | 2 | 1% |
| 3ffe:8041:2:1:290:27ff:fe43:121f | 2 | 1% |
| 3ffe:8114:2000:bf0:2e0:18ff:fe23:cece | 2 | 1% |



### *2010:105:1::/48*

The first two entries in the top 10 source IPv6 source bogons are from the same /48 prefix. Below you will find a table of traffic originated from the bogon prefix 2010:105:1::/48

| proto | srcip | srcport | dstip | dstport | count |
|---|---|---|---|---|---|
| 103 | 2010:105:1::1 | 0 | 2010:104::2 | 0 | 222 |
| 17 | 2010:105:1::3 | 500 | 2010:108:3:3::2 | 500 | 42 |
| 17 | 2010:105:1::2 | 500 | 2010:108::2 | 500 | 4 |

The IPv6 prefix 2010::/16 was mentioned in the draft draft-ietf-ngtrans-6to4 draft. In version 5 of this draft was the prefix changed to 2002::/16. See also:
http://tools.ietf.org/rfcdiff?url2=http://tools.ietf.org/id/draft-ietf-ngtrans-6to4-05.txt

Bogon traffic from 2010:105:1::1 is always between the same two hosts, as you can see in the table above. This PIM (protocol 103) traffic is coming in to the core from one of the SURFnet customers.

## *MORE IPV6 ANALYSES*

The top 10 list of IPv6 bogons is populated by addresses from the ranges 2010::/16 and 3FFE::/16. When we filters those ranges out we end up with just two other ranges, those are addresses from the range: FEC0::/10 and FF02::/16. It turns out that these are actually not real bogons.

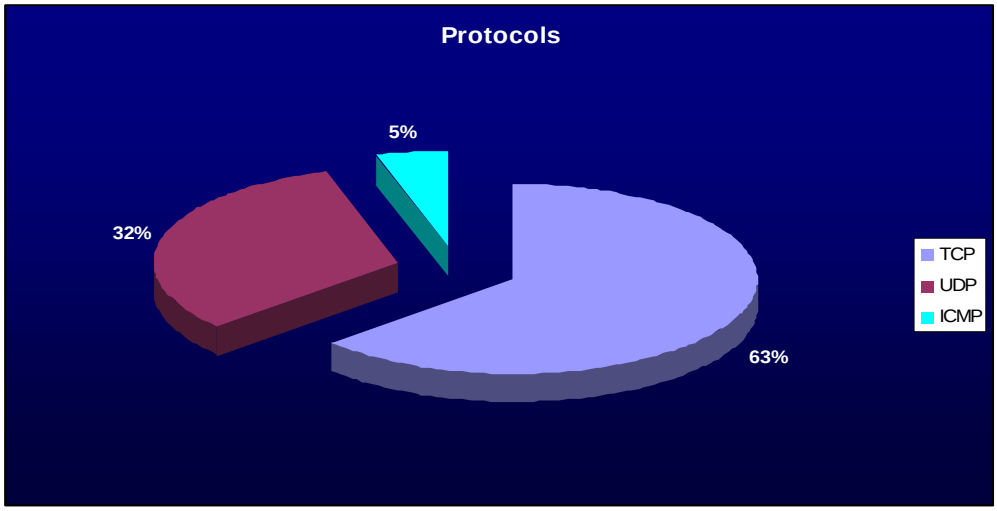
In the netflow data a few flows with a source and destination address from the range FEC0::/10 are seen. These addresses are from the site local range. This range is now officially deprecated, see also RFC 3879.

There are a few IPv6 destination bogons from to the address ff02::16. This address is for the MLDv2-capable routers as described in RFC3810. Because the protocol number in these flows is IPv6-ICMP and the fact that SURFnet6 supports IPv6 multicast this is likely normal MLDv2 traffic and not a bogon.

## PROTOCOL ANALYSES

Below you will find an analysis of what kind of protocols are used by the bogon flows we detected.

| protocol | count | Percentage (%) |
|---|---|---|
| TCP | 15286587 | 63,31844 |
| UDP | 7698538 | 31,88805 |
| ICMP | 1151370 | 4,76908 |
| IPv6 Hop-by-Hop | 2958 | 0,01225 |
| PIM | 1342 | 0,00556 |
| IPv6 | 994 | 0,00412 |
| GRE | 307 | 0,00127 |
| ESP | 256 | 0,00106 |
| IPv6-ICMP | 25 | 0,00010 |
| RSVP | 9 | 0,00004 |
| IGMP | 6 | 0,00002 |
| OSPF | 1 | 0,00000 |




The top3 list is as expected and represents the normal traffic statistics in the network. If we look at other flows we see some interesting things. When we take a closer look at the flows with protocol zero (IPv6 Hop-by-Hop), we see the following:

| proto | srcip | dstip | count |
|---|---|---|---|
| 0 | IPv4 unicast address 1 | 108.122.0.0 | 2948 |
| 0 | IPv4 unicast address 2 | 97.87.6.0 | 5 |
| 0 | IPv4 unicast address 3 | 94.157.6.0 | 2 |
| 0 | IPv4 unicast address 4 | 109.171.6.0 | 1 |
| 0 | IPv6 unicast address 1 | 3ffe:808:0:6240::47 | 1 |
| 0 | 192.168.0.2 | IPv4 unicast address 5 | 1 |

There is one host in the network of one of the Dutch universities which for four days constantly sends traffic to the bogon address 108.122.0.0 with protocol number 0. Protocol number zero is the Hop-by-Hop Options Header which is only used in IPv6. Therefore one would expect IPv6 addresses instead of IPv4 addresses.

More interesting observation were done:

1. Packets with protocol number 4 > IP in IP (encapsulation) [RFC2003]
    - However these had IPv6 addresses in the outer IP header, RFC says explicitly: Version 4.
2 Multiple packets with protocol numbers that are not assigned:
    Protocol: 201
    Protocol: 207
    Protocol: 242
    Protocol: 143

## ROUTING TABLE ANALYSIS

The previous chapters described what kind of bogon traffic was observed when analyzing the netflow data. One of the questions one could ask is if these prefixes are visible in the routing table. During this project the routing tables of the SURFnet border routers as well as the route server on route-views.routeviews.org were periodically checked for bogons. A Perl script was developed to analyze these routing tables. Please see appendix II for the source code. An example output of this script is shown below:

```
andree@kahn:~/projects$ perl bogon-routing-table.pl ford.txt
0.0.0.0/0          0.0.0.0/8
127.0.0.0/8        127.0.0.0/8
192.168.100.0/24   192.168.0.0/16
198.18.0.0/15      198.18.0.0/15
number of prefixes =  203799
number of prefixes with bogon = 4
percentage of prefixes with bogon = 0.00245339771048926 %
```

In this example of the routing table of one of the SURFnet6 border routers, there were two real bogons found, `192.168.100.0/24` and `198.18.0.0/15.` On average between 1 and 5 bogon prefixes were found in the routing tables. When a bogon prefix was found, the technical contact person of the AS number announcing this prefix was contacted. This way we tried to determine whether this was on purpose or if this was a configuration error. Sometimes a reaction was received; these reactions always said that the configuration error had been fixed. However from most of them no reaction was received.

Below is an example of an RFC1918 prefix which was routable for one week.

```
BOFH:~# traceroute-nanog -A -I icmp -n 192.168.100.1
traceroute to 192.168.100.1 (192.168.100.1), 64 hops max, 28 byte packets
 1  145.89.193.250 [AS1103]  0 ms  0 ms  0 ms
 2  145.89.1.41 [AS1103]  1 ms  1 ms  1 ms
 3  145.145.16.5 [AS1103]  2 ms  2 ms  3 ms
 4  145.145.80.10 [AS1103]  2 ms  2 ms  3 ms
 5  195.69.144.94 [AS1200]  2 ms  2 ms  2 ms
 6  63.223.28.161 [AS19151]  91 ms  91 ms  91 ms
 7  63.223.28.201 [AS19151]  91 ms  91 ms  91 ms
 8  63.223.28.141 [AS19151]  94 ms  94 ms  95 ms
 9  63.223.0.65 [AS19151]  123 ms  121 ms  124 ms
10  63.223.20.53 [AS19151]  173 ms  173 ms  173 ms
11  66.186.192.94 [AS19654]  173 ms  173 ms  174 ms
12  63.223.30.134 [AS19151]  165 ms  165 ms  165 ms
13  192.168.100.1 [<NONE>]  165 ms  165 ms  165 ms
```

In the traceroute output above the AS numbers are also shown. This is to show how many organizations are actually routing this prefix. Although it is considered as common best practice to filter on these RFC1918 prefixes, apparently quite some networks are not doing this.

## EMAIL ANALYSIS

One of the scenarios for spammers could be to insert a certain (bogon) prefix in the global routing table, then use an IP address from this prefix to sent spam and when done sending spam mail stop announcing the prefix. This way it would be very difficult to find the real source of the spammer. To see if this is actually happening we searched the netflow data for proof. When analyzing the bogon netflow data for flows with TCP port 25 the result is mostly flows to RFC 1918 addresses. This could be because of misconfigured MX records. When ignoring the RFC1918 addresses and focusing on flows which have a bogon source address there are a small number of flows left which have a destination of TCP port 25.

It turns out that in the different datasets we have found just a few flows from a bogon source address to an SMTP server, ie. destination port 25. Below are the flows which have a bogon source address and port 25 as destination port.

| Bogon source IP | dstport | count |
|---|---|---|
| 180.200.222.44 | 25 | 3 |
| 192.0.2.112 | 25 | 2 |
| 100.100.100.100 | 25 | 2 |
| 3ffe:805::230:48ff:fe11:9956 | 25 | 1 |
| 173.145.21.1 | 25 | 1 |
| 100.39.8.98 | 25 | 1 |
| 93.0.0.6 | 25 | 1 |
| 176.64.100.19 | 25 | 1 |

The table above shows that there is minimal proof that this happened during the three weeks of netflow data. However as seen in the table above there are some flows which match the criteria, this means that some attempts have been made. When looking closer to these flows, we see that these TCP flows were flows with the RST or SYN flag set, this means according to the flow information these TCP sessions never got to established mode.

The RIS[5] project at RIPE was used to check if the prefix was announced during the period of time the SMTP flow was detected. The prefixes were visible in the routing during the period the flows were detected.

It should be mentioned that although we only found minimal proof, this does not mean it does not happen on a larger scale. Picking a prefix from the IANA pool could be too obvious; therefore it would make more sense to pick a prefix from a RIR pool or to hijack an existing prefix. A conclusion about whether this is happening or not and on what kind of scale really depends on the definition of a bogon.

[5] http://www.ris.ripe.net/

In addition to analyzing the netflow information for bogon traffic to TCP port 25, we also analyzed the headers of email messages. A script was written to scan mailheaders for bogon IP addresses. In this case RFC1918 addresses and 127.0.0.1 were not considered as a bogon. The script is written in Perl and the source code can be found in appendix III

Over 84000 emails have been scanned for bogon addresses, about 53000 of these messages were marked as spam. The result of this scan is that almost 5% of all the email marked as spam (by mail servers) contained one or more IPv4 bogon addresses. There were also emails detected with invalid IPv4 addresses in the headers, these addresses looked something like: 111.111.111.278. This shows that some of the headers are forged and not all the IP addresses in the mail header were actually used for relaying.

| | **Non spam email *** | **Spam email **** |
|---|---|---|
| total email scanned | 31502 | 53195 |
| emails with bogon | 7 | 2438 |
| email with bogon % | 0,02% | 4,5% |
| total bogons | 7 | 2762 |

** non spam email = Not marked as spam by mailserver. In this case email from NANOG and IETF mailing lists*
*** Spam email = email marked as spam by mail server (SpamAssassin )*

A plugin for spamassassin has been written. This plugin will scan emails for bogon IP addresses in the "received from" headers. This will improve the spam detection when using spamassassin. The plugin code can be found in Appendix IV

# CONCLUSION

In the three weeks we received the sampled netflow data from the SURFnet6 IP networks, we filtered out over 80 million bogon flows. The majority of these flows are IPv4 flows. About 80 percent of all the destination and 31 percent of all the source IPv4 bogon flows are RFC1918 addresses. During this project we have concentrated on the bogon flows without RFC1918 addresses. We have tried to understand what kind of traffic this is and where it is coming from. A lot of this traffic are scans to a few hosts or specific IP ranges. Sometimes these are scans to a range of well known Windows services; sometimes it is a scan to just one port on a complete network range. Sometimes we were able to identify flows which were the result of most likely configuration errors. At the time of analyzing the flows the bogon prefixes which were analyzed were not in the routing table, so it was difficult to find the source of the bogon prefix. Most of the IPv6 bogon data was traffic from or to 6BONE address space. This can be considered as bogon since 6-6-2006.

One of the conclusions is that there is quite a lot of bogon traffic in the network. However it should be noted that the way how netflow is implemented on the Avici routers is that netflow export will go first before ACL's. This means that if a packet would get dropped by an ACL it would be recorded in the netflow data anyway. As a result not all the traffic in the netflow database was routed into the SURFnet6 core.

When we did an analyses on the protocols used in the bogon flows, we saw that the top 99% consists of TCP, UDP and ICMP. This is a fairly normal representation of protocol usage in a network. When we zoomed in on the 1% other protocols a few flows were found with protocol numbers which are not assigned yet. A few flows had protocol numbers and an inet family mismatch, an example of this is flows with protocol number 0 (zero) which is the IPv6 Hop-by-Hop Option protocol; in these flows IPv4 addresses were used.

During the project we searched for bogons in the routing tables of the SURFnet6 border routers. The conclusion of this is that on average there are always a few bogons in the routing table. The number of bogons found in the routing table will depend on the bogon definition. We contacted all the AS's that were advertising a bogon prefix. In just a few cases a reaction was received, in all of those cases it was mentioned that this was a configuration error.

During this project we tried to identify flows from bogon source addresses towards an SMTP server. This could be a scenario used by spammers when sending spam from a bogon source address. In the 80 million bogon flows analyzed, 12 of these flows were found. Although this is quite a small number of flows, it shows that there are attempts to sent email from bogon addresses towards SMTP servers. More analyses were done on spam email messages. Over 84.000 emails were analyzed for bogon IP addresses in the received from headers. Around 53.000 of these emails were spam messages, in 4,5% of these spam emails bogon IP addresses were found in the headers. Around 31.000 non spam emails were scanned, in 0,02% of these non spam emails bogon IP addresses were found in the headers. So almost 1 in 20 spam emails seems to have a bogon IP address in the "received from" headers. A spamassassin plugin has been written to filter out these headers (see appendix IV).

From a security perspective it is advised to do at least some basic filtering on bogons. In the SURFnet6 network this is realized by implementing anti-spoofing filters on the interfaces connected to the customer as defined in RFC2827 / BCP 38 "Network Ingress Filtering"[6]. On the border routers on the interfaces towards other AS's Unicast RPF (Reverse Path Forwarding) should be applied. In addition to that access lists and BGP filters which filter out the bogon prefixes are recommended. Depending on which bogon definition is used to build these filters, this can be labor intensive task.

[6] http://www.faqs.org/rfcs/rfc2827.html

.

## APPENDIX I

These are the filters used in flowd to store only bogon traffic.

```
# Flow filtering rules
accept quick src 0.0.0.0/8
accept quick dst 0.0.0.0/8
accept quick src 1.0.0.0/8
accept quick dst 1.0.0.0/8
accept quick src 2.0.0.0/8
accept quick dst 2.0.0.0/8
accept quick src 5.0.0.0/8
accept quick dst 5.0.0.0/8
accept quick src 7.0.0.0/8
accept quick dst 7.0.0.0/8
accept quick src 10.0.0.0/8
accept quick dst 10.0.0.0/8
accept quick src 23.0.0.0/8
accept quick dst 23.0.0.0/8
accept quick src 27.0.0.0/8
accept quick dst 27.0.0.0/8
accept quick src 31.0.0.0/8
accept quick dst 31.0.0.0/8
accept quick src 36.0.0.0/8
accept quick dst 36.0.0.0/8
accept quick src 37.0.0.0/8
accept quick dst 37.0.0.0/8
accept quick src 39.0.0.0/8
accept quick dst 39.0.0.0/8
accept quick src 42.0.0.0/8
accept quick dst 42.0.0.0/8
accept quick src 49.0.0.0/8
accept quick dst 49.0.0.0/8
accept quick src 50.0.0.0/8
accept quick dst 50.0.0.0/8
accept quick src 92.0.0.0/8
accept quick dst 92.0.0.0/8
accept quick src 93.0.0.0/8
accept quick dst 93.0.0.0/8
accept quick src 94.0.0.0/8
accept quick dst 94.0.0.0/8
accept quick src 95.0.0.0/8
accept quick dst 95.0.0.0/8
accept quick src 96.0.0.0/8
accept quick dst 96.0.0.0/8
accept quick src 97.0.0.0/8
accept quick dst 97.0.0.0/8
accept quick src 98.0.0.0/8
accept quick dst 98.0.0.0/8
accept quick src 99.0.0.0/8
accept quick dst 99.0.0.0/8
accept quick src 100.0.0.0/8
accept quick dst 100.0.0.0/8
accept quick src 101.0.0.0/8
accept quick dst 101.0.0.0/8
accept quick src 102.0.0.0/8
accept quick dst 102.0.0.0/8
accept quick src 103.0.0.0/8
accept quick dst 103.0.0.0/8
accept quick src 104.0.0.0/8
accept quick dst 104.0.0.0/8
accept quick src 105.0.0.0/8
accept quick dst 105.0.0.0/8
accept quick src 106.0.0.0/8
accept quick dst 106.0.0.0/8
accept quick src 107.0.0.0/8
accept quick dst 107.0.0.0/8
accept quick src 108.0.0.0/8
accept quick dst 108.0.0.0/8
```

```
accept quick src 109.0.0.0/8
accept quick dst 109.0.0.0/8
accept quick src 110.0.0.0/8
accept quick dst 110.0.0.0/8
accept quick src 111.0.0.0/8
accept quick dst 111.0.0.0/8
accept quick src 112.0.0.0/8
accept quick dst 112.0.0.0/8
accept quick src 113.0.0.0/8
accept quick dst 113.0.0.0/8
accept quick src 114.0.0.0/8
accept quick dst 114.0.0.0/8
accept quick src 115.0.0.0/8
accept quick dst 115.0.0.0/8
accept quick src 116.0.0.0/8
accept quick dst 116.0.0.0/8
accept quick src 117.0.0.0/8
accept quick dst 117.0.0.0/8
accept quick src 118.0.0.0/8
accept quick dst 118.0.0.0/8
accept quick src 119.0.0.0/8
accept quick dst 119.0.0.0/8
accept quick src 120.0.0.0/8
accept quick dst 120.0.0.0/8
accept quick src 127.0.0.0/8
accept quick dst 127.0.0.0/8
accept quick src 169.254.0.0/16
accept quick dst 169.254.0.0/16
accept quick src 172.16.0.0/12
accept quick dst 172.16.0.0/12
accept quick src 173.0.0.0/8
accept quick dst 173.0.0.0/8
accept quick src 174.0.0.0/8
accept quick dst 174.0.0.0/8
accept quick src 175.0.0.0/8
accept quick dst 175.0.0.0/8
accept quick src 176.0.0.0/8
accept quick dst 176.0.0.0/8
accept quick src 177.0.0.0/8
accept quick dst 177.0.0.0/8
accept quick src 178.0.0.0/8
accept quick dst 178.0.0.0/8
accept quick src 179.0.0.0/8
accept quick dst 179.0.0.0/8
accept quick src 180.0.0.0/8
accept quick dst 180.0.0.0/8
accept quick src 181.0.0.0/8
accept quick dst 181.0.0.0/8
accept quick src 182.0.0.0/8
accept quick dst 182.0.0.0/8
accept quick src 183.0.0.0/8
accept quick dst 183.0.0.0/8
accept quick src 184.0.0.0/8
accept quick dst 184.0.0.0/8
accept quick src 185.0.0.0/8
accept quick dst 185.0.0.0/8
accept quick src 186.0.0.0/8
accept quick dst 186.0.0.0/8
accept quick src 187.0.0.0/8
accept quick dst 187.0.0.0/8
accept quick src 192.0.2.0/24
accept quick dst 192.0.2.0/24
accept quick src 192.168.0.0/16
accept quick dst 192.168.0.0/16
accept quick src 197.0.0.0/8
accept quick dst 197.0.0.0/8
accept quick src 198.18.0.0/15
accept quick dst 198.18.0.0/15
accept quick src 223.0.0.0/8
accept quick dst 223.0.0.0/8
accept quick src 224.0.0.0/3
accept quick dst 224.0.0.0/3
discard dst 0.0.0.0/0
discard src 0.0.0.0/0
accept quick src 3ffe::/16
#
```

```
#Inet6 rules
accept quick dst 3ffe::/16
discard quick dst 2001:500::/30
discard quick src 2001:500::/30
accept quick src 2001:db8::/32
accept quick dst 2001:db8::/32
discard quick src 2001::/16
discard quick dst 2001::/16
discard quick dst 2002::/16
discard quick src 2002::/16
discard quick src 2003::/16
discard quick dst 2003::/16
discard quick dst 2400::/16
discard quick src 2400::/16
discard quick dst 2404::/22
discard quick src 2404::/22
discard quick dst 2600::/12
discard quick src 2600::/12
discard quick dst 2620::/23
discard quick src 2620::/23
discard quick dst 2a00::/16
discard quick src 2a00::/16
discard quick dst 2a01::/16
discard quick src 2a01::/16
accept src 0::/0
accept dst 0::/0
```

# APENDIX II

```
#!/usr/bin/perl
use Net::Netmask;
use warnings;
use strict;

if ($#ARGV != 0) {
        print "Please specifile file with routing table as argument, for example
        /home/andree/routing-table\n";
        exit;
};

my $file = $ARGV[0];    #

my $routes = 0;
my $bogonroutes = 0;

my @prefixes;
# http://www.cymru.com/Documents/bogon-bn-nonagg.txt
open(MYINPUTFILE, "<bogon-bn-nonagg.txt");
while(<MYINPUTFILE>)
{
        my($line) = $_;
        chomp($line);
        push(@prefixes, $line);
        #$hash{$line};
}
close(MYINPUTFILE);

for my $b (@prefixes) {
        my $x = new Net::Netmask ($b);
        $x->storeNetblock();
}



        open(TABLE, "<$file") || die "Can't open spamfile $file $!\n";
        while (<TABLE>) {
                my $ip;
                #print "$_";
                if ($_ =~ /(\d+\.\d+\.\d+\.\d+)\/(\d+)/) {
                        $routes++;
                        $ip = $1;
                        my $mask = $2;
                        #print "checking $ip \/$2\n";
                        if ($ip =~ /(\d+)\.(\d+)\.(\d+)\.(\d+)/) {
                                if ( (($1 >= 0) && ($1 <= 255)) && (($2 >= 0) && ($2 <= 255)) &&
(($3 >= 0) && ($3 <= 255)) && (($4 >= 0) && ($4 <= 255)) ) {
                                        if(defined(findNetblock($ip))) {
                                                my $localhost = Net::Netmask->new("127.0.0.1");
                                                if (!$localhost->match($ip)) {
                                                        print "$ip/$mask\t" . findNetblock($ip) .
"\n";
                                                        $bogonroutes++;
                                                }
                                        }
                                }else {
                                        print "NOTE: invalid IP address! $ip\n";
                                }
                        }
                }
        }
        close(TABLE);
print "number of prefixes =  $routes\n";
print "number of prefixes with bogon = $bogonroutes\n";
my $percentage = ($bogonroutes / $routes ) * 100 ;
print "percentage of prefixes with bogon = $percentage %\n";
```

# APPENDIX III

source perl script spam mail

```
#!/usr/bin/perl
use Net::Netmask;
use warnings;
use strict;

my $mail = 0;
my $mailwithbogon = 0;
my $bogons = 0;
my $emails = 0;

if ($#ARGV != 0) {
        print "Please directory with spammail as argument, for example
        /home/andree/spammail\n";
        exit;
};

my $dir = $ARGV[0];    #

my @prefixes;
#make sure to remove RFC1918 space from the bogon list!!!
open(MYINPUTFILE, "<bogon-bn-nonagg.txt");
while(<MYINPUTFILE>)
{
        my($line) = $_;
        chomp($line);
        push(@prefixes, $line);
        #$hash{$line};
}
close(MYINPUTFILE);

for my $b (@prefixes) {
        my $x = new Net::Netmask ($b);
        $x->storeNetblock();
}

while (defined($mail = <$dir/*>)) {
        open(SPAM, "<$mail") || die "Can't open spamfile $mail $!\n";
        $emails++;
        my $foundbogon = 'no';
        while (<SPAM>) {
                my $ip;
                #print "$_";
                if ($_ =~ /Received:.+\[(\d+\.\d+\.\d+\.\d+)\]\)/) {
                        $ip = $1;
                        #print "checking $ip\n";
                        if ($ip =~ /(\d+)\.(\d+)\.(\d+)\.(\d+)/) {
                                if ( (($1 >= 0) && ($1 <= 255)) && (($2 >= 0) && ($2 <= 255)) &&
(($3 >= 0) && ($3 <= 255)) && (($4 >= 0) && ($4 <= 255)) ) {
                                        if(defined(findNetblock($ip))) {
                                                my $localhost = Net::Netmask->new("127.0.0.1");
                                                if (!$localhost->match($ip)) {
                                                        print "$ip\t" . findNetblock($ip) . "\n";
                                                        $bogons++;
                                                        $foundbogon = 'yes';
                                                }
                                        }
                                }

else {
                                        print "NOTE: invalid IP address! $ip\n";
                                        $bogons++;
                                        $foundbogon = 'yes';
                                }
                        }
                }
        }
        close(SPAM);
        if ($foundbogon eq 'yes') {
                $mailwithbogon++;
        }
}
print "number of email =  $emails\n";
print "number of email with bogon = $mailwithbogon\n";
print "number of total bogons = $bogons\n";
```

# APPENDIX IV

Below is the source code for the spamassassin plugin. There are 2 files:

1. /etc/spamassassin/bogonreceivedline.cf
2. /etc/spamassassin/bogonreceivedline.pm
   * the exact path maybe different depending on your spamassassin installation

This plugin is written by Bas Toonk (bas<at>toonk.nl) and Andree Toonk (andree<at>sara.nl)

## BOGONRECEIVEDLINE.CF

```
loadplugin      BogonReceivedLine bogonreceivedline.pm
header          BOGONRECEIVEDLINE eval:bogonreceivedline()
describe        BOGONRECEIVEDLINE Check for begon ip header lines
tflags          BOGONRECEIVEDLINE net
score           BOGONRECEIVEDLINE 1
```

## BOGONRECEIVEDLINE.PM

```
package BogonReceivedLine;

use strict;
use Mail::SpamAssassin;
use Mail::SpamAssassin::Plugin;
use Net::Netmask;
our @ISA = qw(Mail::SpamAssassin::Plugin);

sub new {
  my ($class, $mailsa) = @_;
  $class = ref($class) || $class;
  my $self = $class->SUPER::new($mailsa);
  bless ($self, $class);
  $self->register_eval_rule("bogonreceivedline");
  return $self;
}

sub bogonreceivedline {

  my ($self, $permsgstatus) = @_;

  # if a user set dns_available to no we shouldn't be doing MX lookups
  #return 0 unless $permsgstatus->is_dns_available();

  my $hits = 0;
  my $received = $permsgstatus->get("Received");

  my @recs = split('\n',$received);

  for (@recs) {
    my $ip;
    if ($_ =~ /.+\[(\d+\.\d+\.\d+\.\d+)\]\)/) {
      $ip = $1;
      if(defined(findNetblock($ip))) {
        my $localhost = Net::Netmask->new("127.0.0.1");
        if (!$localhost->match($ip)) {
          #print STDERR "$ip\t" . findNetblock($ip) . "\n";
          $hits = 1;
        }
      }
    }
    #print STDERR "ERROR: $_\n";
  }
```

```
  return 1 if $hits;
  return 0;
}

my @prefixes;
# download http://www.cymru.com/Documents/bogon-bn-nonagg.txt
# make sure to remove RFC1918 space from the bogon list!!!
open(MYINPUTFILE, "</etc/spamassassin/bogon-bn-nonagg.txt");
while(<MYINPUTFILE>)
{
        my($line) = $_;
        chomp($line);
        push(@prefixes, $line);
        #$hash{$line};
}
close(MYINPUTFILE);

for my $b (@prefixes) {
        my $x = new Net::Netmask ($b);
        $x->storeNetblock();
}


1;
```